OIPE JC10 JUL 0 2 2004 PATENT & TRADEMARK OFFICE

AF/2126 IFW

PTO/SB/21 (02-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/739,516
Filing Date	12/18/2000
First Named Inventor	Lewis
Art Unit	2126
Examiner Name	Lechi Truong
Attorney Docket Number	US000345

Total Number of Pages in This Submission: 54

ENCLOSURES *(Check all that apply)*

- [x] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
  - [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) ____________
- [ ] After Allowance communication to Technology Center (TC)
- [ ] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [x] Other Enclosure(s) (please Identify below):

postcard

Remarks

Appeal Brief + 2 copies

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Anne E. Barschall, Reg. # 31,089
Signature	A E Barschall
Date	6/29/04

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Typed or printed name	Anne E. Barschall		
Signature	A E Barschall	Date	6/30/04

PTO/SB/17 (10-03)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2004

Effective 10/01/2003. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 330.00

Complete if Known	
Application Number	09/739,516
Filing Date	12/18/2000
First Named Inventor	Lewis
Examiner Name	L. Truong
Art Unit	2126
Attorney Docket No.	US 000345

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☑ Deposit Account:

Deposit Account Number: 14-1270

Deposit Account Name:

The Director is authorized to: *(check all that apply)*

☑ Charge fee(s) indicated below ☑ Credit any overpayments

☐ Charge any additional fee(s) or any underpayment of fee(s)

☐ Charge fee(s) indicated below, **except for the filing fee**

to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	770	2001	385	Utility filing fee	
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) ($)

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

			Extra Claims		Fee from below		Fee Paid
Total Claims		-20** =		X		=	
Independent Claims		- 3** =		X		=	
Multiple Dependent						=	

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claim, if not paid
1204	86	2204	43	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($)

**or number previously paid, if greater; For Reissues, see above*

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	475	Extension for reply within third month	
1254	1,480	2254	740	Extension for reply within fourth month	
1255	2,010	2255	1,005	Extension for reply within fifth month	
1401	330	2401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	330
1403	290	2403	145	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive - unavoidable	
1453	1,330	2453	665	Petition to revive - unintentional	
1501	1,330	2501	665	Utility issue fee (or reissue)	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))	
1801	770	2801	385	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) ________

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 330.00

SUBMITTED BY				(Complete *(if applicable)*)	
Name *(Print/Type)*	Anne E. Barschall	Registration No. (Attorney/Agent)	31,089	Telephone	914-332-1019
Signature	A. Barschall			Date	6/29/04

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

This collection of information is required by 37 CFR 1.17 and 1.27. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application Ser. No.: 09/739,516 Group Art Unit: 2126

Filing Date: 12/18/2000 Examiner: L. TRUONG

Attorney Docket Number US 000345 Inventor Name(s): LEWIS

Title:

Mail Stop Appeal Brief
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

APPEAL BRIEF

Sir:

This is an appeal from the final rejection of Claims 1-11.

I. REAL PARTY IN INTEREST

The real party in interest is Koninklijke Philips Electronics, N.V., a corporation of the Netherlands.

II. RELATED APPEALS AND INTERFERENCES

Applicant is not aware of any related appeals or in interferences.

III. STATUS OF CLAIMS

Claims 1-7 stand rejected under 35 USC 103 over Swenson (US Pat. No. 4,413,317) in view of allegedly admitted prior art.

Claims 8 and 11 stand rejected under 35 USC 103 over allegedly admitted prior art in view of Zhou (5,995,511).

Claims 9-10 stand rejected under 35 USC 103 over allegedly admitted prior art in view of Zhou and further in view of the Mosberger article.

IV. STATUS OF AMENDMENTS

There were remarks, but no amendment under rule 116. Accordingly, there is no unentered amendment

V. SUMMARY OF THE INVENTION

The invention relates to the field of object-oriented programming. The application contains extensive background as to what object-oriented programming is on pages 1-5. The application also defines the term "object" especially on page 2, lines 12 et seq.

> Each object within an O-O system is defined by an interface and an implementation. A software client external to an object depends completely on its interface and not the details of its implementation. The implementation of an object provides the mechanisms and the details that define its behavior. O-O programs are collections of objects that relate to each other through their interfaces.
>
> In a sense, each object is a "black box." Its interface consists of messages that the black box sends and receives. _*Objects actually*_

> *contain code (sequences of computer instructions) and data (information which the instructions operate on)*. Traditionally, code and data have been kept apart. For example, in the C language, units of code are called "functions," while units of data are called "structures." Functions and structures are not formally connected in C. A C function can operate on more than one type of structure, and more than one function can operate on the same structure. This is not true for O-O software. In O-O programming, code and data are merged into a single indivisible thing -- an object. A programmer using an object should not need to look at the internals of the object once the object has been defined. ***All connections with the object's internal programming are accomplished via messages; i.e., the object's interface.*** [emphasis added]

It can be seen from this definition that an object is not the same as data or a data structure. A data structure is something that contains data to be used by code. The object can contain data structures, but those data structures will only be accessible to that object using that object's code. Data structures, unlike objects, typically do not send messages. Data structures are typically passive.

It can also be seen from this definition that an object is not the same as a piece of hardware. An object is data and code, independent of particular hardware implementations. The

fact that an object is represented as a block in a figure does not make it a hardware device. Thus processing objects are not processors. Processors are hardware devices.

Independent claim 1 recites interactions between path objects, data objects, and processing objects. These objects are illustrated in figures 2 and 3. Queue identifiers are stored in a path object, e.g. 115. A data object 110 is received and processed in a processing object 100. A queue 142 corresponding to a second processing object 120 is identified responsive to an indicator corresponding to the data object 110. The data object is then placed in the identified queue.

Independent claim 4 recites performing a process on a data part of a first object 110 by a first processing object 110. A first queue 142, 143 is identified using an indicator part "pointer" of the first data object. Then the indicator part is modified to produce a second data object. The process is then performed on the second data object. Then a second queue to which said second data object is to be transferred is identified.

Independent claim 6 recites a pipeline software architecture. In this architecture, data objects 110 are transferred from a first processing object 100 to a selected one of second and third processing objects 120,130, by queuing the data objects in a queue 142, 143 of the selected one. The architecture defines a path object 115 corresponding to each of the data objects 110. The path object contains an indicator 181,182 of at least one of the second and third processing objects 120, 130. The first processing object 100 defines a process. The result of this process is to insure that a first data object 110 is placed in a queue 142, 143 of one of the second and third processing objects 120, 130, responsively to the path object 115 corresponding to the data object 110.

Independent claim 8 explicitly recites an object oriented programming environment. In this environment, a data object 110 is maintained in a first queue 141 according to a queue indicator associated with the data object. Responsive to the queue indicator, the processing object 100 processes the data object 110. Responsive to the processing, the queue indicator is changed to indicate a second queue 142, 143 destined for a second processing object 120, 130.

VI. THE ISSUES

Are the art rejections correct?

VII. GROUPING OF THE CLAIMS

The claims do not stand or fall together.

VIII. THE ARGUMENT

General comments with respect to the Swenson reference

Swenson is an incredibly enormous reference (99 sheets of drawing, 188 columns of text) relating to managing disk drives. Applicant can find no teaching or suggestion of any object-oriented programming in Swenson. Given the enormity of this reference, Applicant has confined his consideration to the portions pointed to by the Examiner, except as indicated.

The Examiner takes portions out of Swenson context, mischaracterizes them as "objects" when they are not, and then cobbles them together with impermissible hindsight using

Applicant's claims as a road map[1]. Moreover, the Examiner ignores Applicant's definition of "object," denying Applicant's right to be his own lexicographer.

So-called "admitted prior art" (APA)

The Examiner calls the portion of Applicant's specification designated as background "admitted prior art." In fact, Applicant does not admit that each and every definition and explanation of this section is actually prior art. Certainly, object-oriented programming existed prior to the invention; however, it is not at all clear which portions of the definition section were actually in the prior art. The mere placement of this section at the front of the application does not in and of itself constitute an admission that it is all actually prior art. Accordingly, the Examiner has not made a *prima facie* case with respect to those portions of the claims that the Examiner rejects over this definitional section.

Claim 1

Claim 1 stands rejected over Swenson in view of the admitted prior art.

---

[1] The CAFC has said

> The "as a whole" instruction in title 35 prevents evaluation of the invention part by part. Without this important requirement, an obviousness assessment might break an invention into its component parts (A + B + C), then find a prior art reference containing A, another containing B, and another containing C, and on that basis alone declare the invention obvious. This form of hindsight reasoning, using the invention as a roadmap to find its prior art components, would discount the value of combining various existing features way to achieve a new result – often the very definition of invention. Ruiz v. A. B. Chance Co., http://www.ll.georgetown.edu/federal/judicial/fed/opinions/03opinions/03-1333.html at p. 7, 357 F. 3d 1270, 2004 US App. Lexis 1325, 69 U.S.P.Q. 2d (BNA) 1686 (Fed. Cir 2004)

The Examiner first points to column 1, lines 33-45 and col. 2, ll. 20-45 as showing transfer of data. However claim 1 does not recite transfer of data. Claim 1 recites a data object. Mere data fails to teach or suggest a data object, as explained above.

Moreover, in Swenson, data is transferred to host processors. Host processors fail to teach or suggest processing <u>objects</u>. <u>Objects</u> are conglomerations of code and data, as defined in Applicant's specification. Processors are physical hardware items.

Claim 1 recites storing queue identifiers in a path object. The Examiner then points to queue identifiers in a table – col. 162:ll 1-10; col. 2, ll. 21-50; col. 188, ll. 11-21. A table fails to teach or suggest an object. A table is a data structure that contains only data. An object contains data and code. A table is directly accessible from the outside by software. An object communicates only using messages, with its internals not being accessible from the outside.

The command queue store of column 2 appears to be only a storage device, again with its contents directly accessible by external software.

Claim 1 recites path objects. The Examiner also points to paths col. 2, ll. 4-50; col. 1, ll. 9-62. However paths fail to teach or suggest path objects. A path "exists between the storage control unit and the processor and there are not connection paths between at least one of the storage control units and at least one of the processors," per column 1. Again, as the reference uses the word "path," what is meant is a hardware connection between hardware devices. An object is data and code.

The Examiner says that the reference has a "data object," pointing to:

- the status of the storage unit col. 1, ll. 15-50; but status of a hardware storage unit is data, not a data object;

- reporting status col. 1, ll. 55-68; but reporting status is sending a message. A message is not a data object;
- command queue for execution, col. 161, ll. 1-50; col. 188, ll. 9-67. A command queue is a data structure, not a data object.

The Examiner purports to find processing objects in the reference, as follows:

- storage unit, col. 1, ll. 55-68; but storage units here appear to be hardware devices, while an object is data and code;
- SCU, col. 161, ll. 1-60; The SCU here is supposed to be in figure 93, which Applicant cannot find in the reference[2]; however, Applicants believe — based on reading the text — that all the things that Swenson calls "unit" in this section are actually hardware devices, not "objects";
- Host processor, col. 1, ll. 55-68, col. 188, ll. 9-62 – again a hardware device, not an object;
- Specified host, col. 161, ll. 1-60 – again referring to figure 93, which Applicant can't find, but apparently also a hardware device, not an object.

Accordingly, the Examiner has failed to make a *prima facie* case against claim 1.

<u>Claims 4 & 6</u>

While the recitations of claims 4 & 6 are different and separately patentable from those of

---

[2] Curiously, the last sheet of drawing has figures 91b, 92 & 94. The previous sheets have 90 & 91a. Applicant wonders if there may possibly have been some printing error in this patent.

claim 1, as set forth above in the summary, the rejection is similarly defective. Again the Examiner confuses

- data with a data object;
- a data structure with a data object;
- a path with a path object; and
- a processor or storage unit with a processing object,

ignoring the definition of "object" as set forth in Applicant's specification. These distinctions have been discussed at great length above. Given these errors in interpretation of the reference, it is impossible to determine how the reference may apply to the claims. As a result, the Examiner has failed to present a prima facie case against claims 4 and 6.

<u>Claim 7</u>

Claim 7 recites that the process [in the first processing object] generates an indication of a result. In the preferred embodiment, this may be a normal indication invoking 181 or an error indication invoking 182. The first data object is placed in a queue 142, 143 of one of other processing objects 120, 130. This placement is responsive to the path object 115, 181, 182 and the indication.

In other words, as shown in the preferred embodiment, with respect to fig. 2, the successful or unsuccessful termination of the process will choose a section 181 or 182 of the path object 115. This section 181 or 182 will then determine whether the normal processing object queue 142, or the error processing object queue 143 is chosen.

Against these recitations, the Examiner cites col. 36, ll. 5-8 and col. 186, ll. 30-40 of

Swenson. As far as Applicant can tell, these sections relate only to hardware processors, messages, hardware disk drives, and hardware paths. There is no teaching or suggestion of objects. Given these errors in interpretation of the reference, it is impossible to determine how the reference may apply to the claim. As a result, the Examiner has failed to present a prima facie case against claim 7.

Claim 8

Claim 8 is paraphrased in the summary section above.

This claim stands rejected over Applicant's definitional section in view of Zhou. Applicant has previously pointed out the defect in the Examiner's allegation of "admitted prior art;" however, even if, hypothetically, this material were found all to be prior art, the rejection would still be defective.

While Zhou is not quite as long as Swenson, it still has 13 sheets of drawing and 30 columns of text. Accordingly, it is also a complex reference, and Applicant will confine his remarks to those sections pointed to by the Examiner, except as indicated.

The Zhou reference does not relate to an object-oriented programming environment. Zhou relates to a digital network for transmitting messages. Again the Examiner takes bits and pieces out of context and cobbles them together using Applicant's claims as a road map. Those of ordinary skill in the art would not do this; and in fact could not translate any teachings of this reference into an object-oriented programming environment without undue experimentation.

The Examiner appears to interpret the "cells" of Zhou as data objects; however, at col. 3, line 48 -50, the reference indicates that these cells are actually pieces of data packets. This

means that they are data structures, not data objects.

The Examiner points to connection table 52, entry 52(c), and col. 8, ll. 48-67; and col. 5, ll.5-67. This all seems to relate to processing of queues in the conventional context of data structures. There does not appear to be any queue indicator associated with a data object. Data structures fail to teach or suggest data objects.

The Examiner also seems to indicate that the messages are determining their own "destiny." In saying this, the Examiner seems to confuse "destiny" with "destination." Messages are sent to a physical destination in the reference based on their header information, per col. 3, ll 54, et seq. By contrast, in the claims, the queue is for a processing object which will process the data object. The destiny that the data object is choosing is how it will be processed. This *destiny* is not taught or suggested by the physical *destination* of a message.

The Examiner also seems to think that the descriptions of "virtual paths" in the reference somehow make the reference more pertinent to the claims; however, col. 3, l. 54 through col. 4 line 9 of the reference make clear that these virtual paths are still used to achieve physical transmission of a message from a physical source to a physical destination. Again, this language is referring ultimately to a physical path, not a choice of processing object as in the claim.

Also, since the messages in Zhou travel along physical paths, as shown in Fig. 1. This fails to teach or suggest path *objects*.

In fact, the whole rejection of claim 8 seems to be based on similarity of words rather than the true significance of the underlying technology. These superficial word similarities make this rejection is a bit like a series of puns.

Accordingly, Applicant respectfully submits that the Examiner has failed to make a *prima*

facie case against claim 8.

Claim 9

Claim 9 recites that the queue indicator is stored in a path object associated with the data object. The processing [recited as being in the processing object in the independent claim] comprises querying the path object.

Against this recitation, the Examiner makes some very jumbled statements about path X querying path Y. Applicant has not recited anything of the sort. It is not clear how this statement relates to the claim.

The Examiner then refers to a series of data structures and data in Zhou: tables, table entries; and path identifiers. As explained before, data and data structures fail to teach or suggest objects.

Then the Examiner calls up a fourth reference, Mosberger. The number of references here militates against a finding of obviousness.

Moreover, Mosberger seems to be cited only because it contains the term "path object." This term "path object" is used in the context of an operating system with routers. As far as Applicant can discern from the cited section, packets of data are being transmitted between physical destinations. It is not at all clear how this context could relate to the invention. This obscure portion of this reference can only be connected with Applicant's invention by performing a keyword search in the PTO computers using the terms in Applicant's claims. One of ordinary skill in the art would not design in this fashion. The Examiner is again making an impermissible hindsight reconstruction using Applicant's claims as a road map.

IX. CONCLUSION

Applicant respectfully submits that he has answered each issue raised by the Examiner and that the application is accordingly in condition for allowance. Such allowance is therefore respectfully requested.

Respectfully submitted,

By [signature]

Anne E. Barschall
Reg. No. 31,089
(914) 332-1019
fax 914-332-7719
June 29, 2004

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited this date with the United States Postal Service as first class mail in an envelope addressed to

Mail Stop Appeal Brief
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

On 6/30/04 (Mailing Date)
By [signature] (Signature)

X. APPENDIX

1. (original) A method of determining the flow of a data object in a software architecture using queues to organize the transfer of data from one processing object to another, comprising the steps of:

storing queue identifiers in a path object;

receiving and processing a data object in a first of said processing objects;

identifying a queue corresponding to a second of said processing objects responsively to an indicator corresponding to said data object;

placing said data object in a queue identified in said step of identifying.

2. (original) A method as in claim 1, wherein said step of identifying includes determining a result of said step processing.

3.(original) A method as in claim 2, wherein said step of identifying includes determining a result of said step processing and said result corresponding to said queue.

4. (original) A method for determining the flow of data in a software architecture in which queues are used to organize the transfer of data from one process to another process, comprising the steps of:

performing a process on a data part of a first data object, by a first processing object;

identifying a first queue to which said first data object is to be transferred from a indicator

part of said first data object;

modifying said indicator part of said first data object to produce a second data object;

performing said process on said second data object;

identifying a second queue to which said second data object is to be transferred.

5. (original) A method as in claim 4, further comprising determining a result of said step of performing, said step of identifying including identifying said second queue responsively to said step of determining.

6. (original) A pipeline software architecture in which data objects are transferred from a first processing object to a selected one of second and third processing objects by queuing the data objects in a queue of said selected one, comprising:

a definition of a path object corresponding to each of said data objects;

at least one of said path objects containing an indicator of at least one of said second and third processing object;

said first processing object defining a process a result of which is to insure that a first data object processed by said first processing object is placed in a queue of said at least one of said second and third processing objects responsively to one of said path objects corresponding to said first data object.

7. (original) An architecture as in claim 6, wherein said process includes the generation of an indication of a result of a subprocess of said first processing object and said first data object

processed by said first processing object is placed in said queue of said at least one of said second and third processing objects responsively to one of said path objects corresponding to said first data object and responsively to said indication.

8. (previously presented) In an object oriented programming environment, a method comprising executing the following operations in at least one data processing device:

- maintaining a data object in a first queue according to a queue indicator associated with the data object;
- responsive to the queue indicator, processing the data object with a first processing object; and
- responsive to the processing, changing the queue indicator to indicate a second queue destined for a second processing object;

whereby the data object determines its own destiny.

9. (previously presented) The method of claim 8, wherein the queue indicator is stored in a path object associated with the data object and the processing comprises querying the path object.

10. (previously presented) The method of claim 9, wherein the path object includes a table of queue indicators.

11. (previously presented) The method of claim 8, wherein

- the processing comprises determining a normal or faulty outcome state of the data object; and

- the changing is dependent on said normal or faulty outcome state.